

**REMARKS**

Claims 1 and 3-20 were pending at the time of examination. Claims 1, 10, 13, 14, 16 and 17 have been amended. Claim 9 has been canceled. No new matter has been added. The applicants respectfully request reconsideration based on the foregoing amendments and these remarks.

**Claim Rejections – 35 U.S.C. § 112**

Claims 10-13 were rejected under 35 U.S.C. § 112, first paragraph as failing to comply with the written description requirement. In particular the phrase “suitable for” was considered to be indefinite. The applicants have amended claim 10 to recite “operable to compile” and amended claim 13 to recite “operable to store.” The applicants submit that claims 10-13 as amended are definite and that the rejection under 35 U.S.C. § 112 of claims 10-13 be removed.

**Claim Rejections – 35 U.S.C. § 103**

Claims 1 and 3-20 were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,308,320 to Burch (hereinafter “Burch”) in view of U.S. Patent No. 5,408,665 to Fitzgerald (hereinafter “Fitzgerald”). The applicants respectfully traverse these rejections.

Claim 1 has been amended to more clearly define the invention and the distinctions over the Burch and Fitzgerald combination. As was discussed in the previous Office Action response, compilers generally accept a source file containing multiple data structure definitions and associated functions that can operate on the data structures, and produce an object file containing a realization, or instance, for each of those data structures and its functions. The form of those realizations or instances depends not only on the data structure itself, but also on the instructions presented to the compiler. In the course of development, the same source file may be compiled with different instructions into different object files. The Burch invention discloses a mechanism for avoiding producing an object file (embodying both a source and compiler instructions) when such an object file has been produced before.

In modern programming languages, such as C++, there are language constructs known as templates to those of ordinary skill in the art. The templates enable generic description of classes and functions over a range of types or values. The same template and a set of operations that can be performed on the template (e.g., a “stack” data structure with the operations “push,” “pop,” “list,” and so on) may well appear in multiple source files. When these generic templates are specialized on a particular data type (e.g., a “stack” of integers) or value, they are referred to as

instances of the template. A particular instance (e.g. a stack of integers) may very well be present in each of several object files.

The applicants' invention, as defined in claim 1, pertains to a mechanism for avoiding producing multiple instances (e.g., multiple copies of data structures representing a stack of integers) across object files and libraries. In essence, while Burch reduces the number of compilations by checking if an object file already exists in his depository, the applicants' invention operates to reduce the amount of work in each compilation by checking if an instance exists within one or more libraries.

An alternative way of explaining this is that the applicants' invention as defined in claim 1, avoids the creation of multiple instances independently of the instructions given to the compiler. That is, if the same source file is given to the compiler with two different sets of compiler instructions, the applicants' invention, as defined in claim 1, would only create a single instance. In contrast, in Burch, if the same source file was given to the compiler with two different sets of compiler instructions, the Burch system would create two object files, each one having a unique realization. Hence the use of linker symbol names and the reduction of work in the applicants' invention, in contrast to a hash of object file names and compiler instructions used in Burch.

Second, the applicants' invention, as defined in claim 1, is independent of the object structure. While it is generally true, as the Examiner states, that one can extract objects from libraries, it is not true for template instances that are placed within an object file. This contrasts with object files, such as those relied upon in Burch, which are independently extractable from static archive libraries, such as Burch's depository of object files. Furthermore, the Examiner's statement is also irrelevant to the invention as defined in claim 1, because claim 1 does not recite any extraction of objects. On the contrary, the invention as defined in claim 1, merely concerns the recognition of an instance in a library, not the recognition of an object file. Once the instance is identified in a library, no more work needs be done. The instance will appear in the compiled object file as a consequence of using the one or more libraries during compilation.

The applicants are aware that the use of the term "instantiation" is a bit overloaded in computing these days, and that the term "template" may be construed very generically if its context is disregarded. Therefore, the applicants have amended claim 1 to specify that the source program is written in the C++ or the Ada programming language, in which both the terms "instance" and "template" have very well defined meanings to a person of ordinary skill in the art. It is respectfully submitted that neither Burch nor Fitzgerald, alone or in combination, shows

this combination of features, and for at least these reasons, the rejection of claim 1 is unsupported by the art and should be withdrawn.

Claims 3-9 are all dependent from claim 1, and the rejection of these claims is unsupported by the cited art for at least the reasons discussed above with regards to claim 1, and should be withdrawn.

Claim 10 is a system claim having limitations similar to the limitations of claim 1, and has been amended in a similar way. Thus for at least these reasons presented above with respect to claim 1, the rejection of claim 10 is unsupported by the art and should be withdrawn.

Claims 11-13 are all dependent from claim 10, and the rejection of these claims is unsupported by the cited art for at least the reasons discussed above with regards to claim 1, and should be withdrawn.

Claim 14 is a method claim having limitations similar to the limitations of claim 1, and has been amended in a similar way. Thus for at least these reasons presented above with respect to claim 1, the rejection of claim 10 is unsupported by the art and should be withdrawn..

Claims 15-16 both depend from claim 14, and the rejection of these claims is unsupported by the cited art for at least the reasons discussed above with regards to claim 14, and should be withdrawn.

Claim 17 is a *Beauregard* claim corresponding to claim 1. For reasons substantially similar to those set forth above with regards to claim 1, the applicant respectfully contends that the rejection of claim 17 is unsupported by the cited art and should be withdrawn.

Claims 18-20 all depend from claim 17, and are *Beauregard* claims corresponding to claims 3, 4, and 6, respectively. The rejection of these claims is unsupported by the cited art for at least the reasons discussed above with regards to claim 17, and should be withdrawn.

**Conclusion**

The applicants believe that all pending claims are allowable and respectfully request a Notice of Allowance for this application from the Examiner. Should the Examiner believe that a telephone conference would expedite the prosecution of this application, the undersigned can be reached at the telephone number set out below.

Respectfully submitted,  
BEYER WEAVER & THOMAS, LLP

*Fredrik Mollborn*

Fredrik Mollborn  
Reg. No. 48,587

P.O. Box 70250  
Oakland, CA 94612-0250  
(650) 961-8300